GIT AND GITHUB

Overview of Git:-

Git is a powerful and versatile tool for managing project versions and collaborations. To understand its full potential, let's dive into some key concepts:

Repositories:

- Think of a repository as a self-contained snapshot of your project, including all its files and revisions. These can be local on your machine or remote on platforms like GitHub.

Commits:

- At specific points, you "commit" your project's current state, capturing the changes made since the last commit. Each commit has a unique identifier and a descriptive message.

Branches:

- Branches let you experiment with different features or bug fixes without affecting the main project. Think of them as parallel paths within the same repository. You can switch between branches and merge them to integrate changes.

Working Directory:

- This is the local folder where you actively edit and manage your project files. Changes made here are not yet recorded as commits unless you explicitly commit them.

Staging Area:

- Before creating a commit, you stage the specific files you want to include from your working directory. This allows you to choose which changes to capture.

Remotes:

- Remote repositories are copies of your local repository located on platforms like GitHub. You can push your commits to a remote to share them with others and pull updates from their contributions.

Index:

- The index, also called the staging area, is a temporary file that tracks the files you've marked for inclusion in the next commit.

HEAD:

- This is a pointer to the latest commit you made on the current branch. It essentially indicates the version of your project considered "active."

Master Branch:

- This is the default branch in most repositories, typically representing the main stable version of your project. You can create and work on other branches for development and merge them back into master later.

Merging:

- Merging combines changes from different branches into a single branch. This can sometimes involve resolving conflicts if the same files were modified differently in each branch.

Pull Request (PR):

- A pull request is a formal way to propose changes from one branch to another, usually from a feature branch to the main branch. This often involves reviewing and discussing the changes before merging them.

Push and Pull:

- `git push` sends your local commits to a remote repository. `git pull` fetches changes from a remote repository and merges them into your local branch.

Stash:

- Occasionally, you might want to temporarily shelve your current work without committing it. The `git stash` command allows you to store these changes and restore them later without losing any progress.

These are just some of the fundamental Git concepts.

<u>**Basic commands of Git:-**</u>

Here are some basic Git commands to get you started:

Initialization and Configuration:

- git init - Creates a new Git repository in an existing directory.
- git config - Sets global or local configuration options for Git.

Status and Inspection:

- git status - Shows the status of changes in the working directory and staging area.

- git diff - Shows the differences between files in the working directory and the staging area, or between different commits.
- git log - Shows a list of commits made in the repository.

Tracking Changes:

- git add - Adds files or changes to the staging area, preparing them for the next commit.
- git commit - Creates a snapshot of the current state of the repository with a descriptive message.
- git rm - Removes files from the working directory and staging area.

Branching and Merging:

- git branch - Lists, creates, or deletes branches.
- git checkout - Switches to a different branch or restores files from a previous commit.
- git merge - Combines changes from one branch into another.

Remote Repositories:

- git remote - Manages connections to remote repositories.
- git fetch - Downloads changes from a remote repository without merging them.
- git pull - Fetches and merges changes from a remote repository.
- git push - Sends commits from your local repository to a remote repository.

Undoing Changes:

- git reset - Undoes changes to files or commits.
- git revert - Creates a new commit that reverses the changes of a previous commit.

Other Useful Commands:

- git clone - Copies a remote repository to your local machine.
- git stash - Temporarily shelves uncommitted changes for later use.
- git branch --merged - Lists branches that have been merged into the current branch.
- git branch --no-merged - Lists branches that have not been merged into the current branch.

GitHub, GitLab And BitBucket:-

Here's a breakdown of GitHub, GitLab, and Bitbucket :-

GitHub:

- World's largest and most popular Git hosting platform.
- Open-source, with free public repositories and paid plans for private repositories and additional features.
- Known for its extensive community, social coding features, and wide range of integrations with third-party tools.
- Often used for open-source projects, showcasing code, and collaboration.

Key Features:

- Pull requests with code reviews and discussions
- Issue tracking and project management tools
- Wikis for documentation
- Integrations with other developer tools

GitLab:

- Full DevOps lifecycle platform, offering Git repository management, continuous integration/delivery (CI/CD), container registry, and more.

- Open-source, with both free and paid self-hosted options, as well as a cloud-based version.
- Focused on security, compliance, and scalability.
- Often used by enterprises and teams requiring more control and customization.

Key Features:

- Similar features to GitHub, plus:
- Built-in CI/CD pipeline
- Container registry
- Security scanning
- Issue boards for project management

Bitbucket:

- Git hosting platform owned by Atlassian, primarily integrated with other Atlassian tools like Jira and Trello.
- Offers free and paid plans for individuals and teams.
- Known for its focus on enterprise-grade features, security, and integration with the Atlassian suite.
- Often used by teams already using other Atlassian products.

Key Features:

- Pull requests, issue tracking, and project management
- Built-in code review tools
- Integration with Jira and Trello
- Deployments to cloud platforms

Industrial practices of using Git:-

In the professional world, Git plays a crucial role in managing development workflows and fostering collaboration. Here are some key practices adopted by industries to leverage Git effectively:

Branching Strategies:

- Feature Branches: Dedicated branches for individual features or bug fixes, preventing disruption to the main codebase.
- Release Branches: Stable branches representing production releases, ensuring smooth deployment and rollback if needed.
- Hotfix Branches: Quick fixes applied to production branches for critical issues with minimal disruption.

Workflow Standardization:

- Pull Requests: Formal process for reviewing and merging code changes, ensuring quality and clarity.
- Code Reviews: Collaborative assessments of code changes before integration, promoting best practices and identifying potential issues.
- Continuous Integration (CI): Automated testing and build processes triggered by code changes, providing timely feedback and catching regressions early.

Collaboration and Visibility:

- Issue Tracking: Integrates Git with issue tracking systems, linking tasks to code changes and providing context for development efforts.
- Wikis and Documentation: Git repositories can host project documentation, accessible to all team members and promoting knowledge sharing.
- Team Communication: Utilizing channels like pull request discussions and dedicated tools to facilitate effective communication and collaboration around code changes.

Security and Access Control:

- User Roles and Permissions: Granting different access levels to team members based on their roles and responsibilities.
- Branch Protection: Configuring rules to enforce specific workflows and prevent unauthorised modifications to critical branches.
- Code Signing and Verification: Digitally signing code commits to ensure integrity and traceability of changes.

Cloning a repo to local:-

Cloning a Git repository creates a local copy of a remote repository on your computer. This allows you to work on the project, make changes, and push them back to the remote repository when you're done. Here's how to do it:

1. Open a terminal:

On Windows, open the Start menu and search for "Command Prompt" or "Git Bash."

On macOS, open Spotlight and search for "Terminal."

2. Locate the repository URL:

Find the URL of the remote repository you want to clone. It's usually displayed on the repository's webpage. For example, on GitHub, it's located under the "Clone or download" button.

3. Use the git clone command:

In the terminal, type the following command, replacing "<URL>" with the actual URL of the repository:

```
git clone <URL>
```

Press Enter.

4. Observe the cloning process:

The terminal will display messages indicating the cloning progress. It may take some time depending on the size of the repository.

5. Verify the cloned repository:

Once the cloning is complete, you'll see a new folder created in your current directory with the same name as the repository.

You can navigate to this folder and browse the files inside.